FB.MF\_CLASS\_MASKS\_FIELD

Детектор определяет ошибку, в ходе которой класс определяет поле с тем же именем, что и видимое поле экземпляра в суперклассе. Это сбивает с толку и может указывать на ошибку, если методы обновляют или получают доступ к одному из полей, когда им нужно другое.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Ошибка разработчика |
| CWE | — |
| CVE | — |
| Возможные последствия | Логическая ошибка, путаница при доступе к полю, использование неверного поля, сложность в сопровождении кода |
| Комментарий | Подкласс объявляет поле с тем же именем, что и поле в суперклассе, из-за чего оно скрывает родительское поле. Это может привести к неожиданному поведению, когда методы работают с разными данными, в зависимости от контекста обращения. Рекомендуется использовать наследуемое поле или переименовать поле в подклассе, чтобы избежать путаницы. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() {  class Parent {  protected int value = 10;  }  class Child extends Parent {  // Маскировка поля суперкласса  protected int value = 20;  public void printValues() {  System.out.println("Child value: " + value);  System.out.println("Parent value: " + super.value);  }  }  Child child = new Child();  child.printValues();  } | public static void correctTest() {  class Parent {  protected int value = 10;  }  class Child extends Parent {  // Используется унаследованное поле  public void printValues() {  System.out.println("Value: " + value);  }  }  Child child = new Child();  child.printValues();  } |

Методы устранения уязвимости при срабатывании детектора:

1. **Переименуйте поле в подклассе:** Дайте полю в подклассе имя, которое отличается от имен полей в суперклассе. Это самый простой и рекомендуемый способ избежать маскировки и путаницы.
2. **Используйте унаследованное поле:** Если подклассу действительно нужно работать с тем же концептуальным полем, рассмотрите возможность использования унаследованного поля без объявления нового поля с тем же именем.
3. **Будьте внимательны при доступе к полям:** Если маскировка неизбежна (что не рекомендуется), всегда явно указывайте, к какому полю вы обращаетесь (this.fieldName для поля подкласса, super.fieldName для поля суперкласса).
4. **Проанализируйте структуру классов:** Пересмотрите иерархию классов и подумайте, является ли объявление поля с тем же именем в подклассе действительно необходимым. Возможно, существует более удачное решение с точки зрения дизайна.